Available online at www.sciencedirect.com

ScienceDirect

IERI Procedia 4 (2013) 253 – 260

Procedia IERI

www.elsevier.com/locate/procedia

2013 International Conference on Electronic Engineering and Computer Science

S/W Fault-tolerant OFP System for UAVs based on Partition Computing

Eun-Hye Jeong[a], Jung-Guk Kim[b]*

[a]*Hankuk University of Foreign Studies, San-89 Mohyun, Youngin 449-791, Republic of Korea*
[b]*Professor, Hankuk University of Foreign Studies, San-89 Mohyun, Youngin 449-791, Republic of Korea*

Abstract

Partition computing of the new Integrated Modular Avionics architecture reduces the heavy cabling of traditional federated architecture. On the other hand, fault-tolerant Operational Flight Programs (OFP) for unmanned aerial vehicles have usually been implemented as primary-backup systems based on dual nodes. However, in the case of a small UAV, it is preferred to implement a S/W fault-tolerant system that runs primary and recovery systems together in a single flight control computer to reduce the payload. In this case, because the primary and backup must not interfere with each other in using CPU and memory, it is common to use virtualization-based partitions. In this paper, a new S/W fault-tolerant OFP based on the real-time-object partition, TMO.p, is presented to overcome the large overheads of virtualization approaches.

Keywords: Fault-tolerance; OFP; UAV; partition computing; TMO; TMO.p;

1. Introduction

Flight control systems are required to have high reliability and availability in timeliness and fault-tolerance. For timeliness computing, deadline-driven real-time schemes are commonly used. For example, a Time-

* Jung-Guk Kim. Tel.: +82-10-6206-4367; fax: +82-31-339-3443.
E-mail address: jgkim@hufs.ac.kr.

doi:10.1016/j.ieri.2013.11.036

triggered Message-triggered Object (TMO) model [Kim et al., 1994] has been used to implement an Operational Flight Program (OFP) for Unmanned Aerial Vehicles (UAV) [Kim, S. G. et al., 2009; Kim D. H. et al., 2009; Song et al., 2010; Kim J. G., et al., 2011]. A TMO is a real-time active object that has several periodic time-triggered tasks (SpM: Spontaneous Method), and message-triggered tasks (SvM: Service Method), as its member threads. For fault-tolerance, it is common to use Distributed Recovery Blocks (DRB) based on H/W and S/W dualization [Kim, 1989]. The DRB mechanism has been evolved to the Primary Shadow TMO Replication (PSTR) scheme [Kim et al., 1996; Kim J. et al., 2011], to support both timeliness and fault-tolerance. However, in the case of small UAVs, the H/W replicated DRB fault-tolerant system is not preferred because of its heavy payload.

On the other hand, avionics S/W architecture has been evolved to the partition computing of the new Integrated Modular Avionics (IMA) [Morgan, 1991] architecture, to reduce the heavy cabling of the traditional federated/distributed architecture. In partition computing, each partition that is a group of periodic/aperiodic tasks must not interfere with other partitions in using CPU and memory, according to the Avionics Application Standard Software Interface (ARINC) 653 standard [AEEC, 2005]. That is, a partition with a given period and CPU duration must be executed independently from other partitions, in using the H/W resources. The partition computing is usually used to run multiple independent applications, such as a cockpit system, flight-control system and radar system together, in a single Flight Control Computer (FCC).

Since partition computing has been introduced, new partition-based primary/backup OFP approaches have recently been considered. The partition-based approach is a S/W fault-tolerant scheme for a small UAV, in order to reduce the payload, without dualizing the H/W. In this approach, the primary and backup OFP are organized into independent partitions, respectively, in a single FCC. This approach comes from the fact that reduction of payload is much more important than the taking over of H/W faults due to remarkably increased H/W reliability. In implementing partition-based fault-tolerant systems, one possible approach is to use virtualization [Han et al., 2011; VanderLeest et al., 2010]. By running the primary and backup OFP on two separated virtual machines, the primary can run independently from the backup OFP. However, this approach seriously suffers from slow responses and lack of timeliness, due to the H/W emulation overhead.

In this paper, a new S/W fault-tolerant OFP based on a real-time-object partition model, named Time-triggered Message-triggered Object Partition (TMO.p) [Lee, 2012], is presented, to overcome the large overhead of the virtualization approach. TMO.p is an extension of the TMO, and is a model to support real-time-object-based partition computing. Period, duration and initial-offset are given to a TMO.p, which is composed of several SpMs and SvMs. The timing constraints for multiple TMO.p's are given in such a way that any periodic occurrence of a CPU duration does not overlap the durations of other TMO.p's. Also, the memory usage of a TMO.p is restricted to the scope of the object. So a TMO.p may be called a light-weight partition. In this approach, each of the primary and backup OFP is mapped into an OFP-TMO.p. As a real-time TMO.p engine, RT-eCos.p [Lee et al., 2012; Kim J. G. et al., 2009] has been used. RT-eCos.p is a real-time embedded kernel that supports a two-level scheduler, which consists of a partition scheduler and a deadline-driven task scheduler. To verify the system, a Hardware-In-the-Loop Simulation (HILS) system for an unmanned helicopter has also been developed using the open-source FlightGear simulator.

In section 2, the TMO.p model and RT-eCos.p kernel are introduced briefly as related works. In section 3, design and implementation of the TMO.p-based S/W fault-tolerant OFP scheme for unmanned helicopters are described. Finally, simulation results will be discussed, and conclusions will be given.

2. Related Works

In this section, the TMO.p model and embedded kernel RT-eCos.p for supporting TMO.p are briefly introduced.

2.1. TMO.p model

As an extension of the TMO model, TMO.p [Lee, 2012] supports the partition concept. TMO.p is an active-object partition that has several time-triggered and message-triggered tasks as its member threads (Fig. 1). Each TMO.p uses its own non-overlapping periodic CPU slots, and the memory usage is restricted to the object scope. For timeliness-guaranteed computing, a TMO.p instance consists of three types of object members: an ODS (Object Data Store), time-triggered methods (SpM: Spontaneous Method), and message-triggered methods (SvM: Service Method). An SpM is a periodic member-thread that is activated by a pre-given timing constraint, and must finish its periodic executions within the given deadline. An SvM is also a member-thread, which is activated by an event-message from another method. An SvM is also scheduled to finish the message processing within the pre-given deadline. In OFP applications, SpMs are mainly used in periodic sensing, and generating control outputs. SvMs are used in processing request messages from a Ground Control Station (GCS). Besides the timing constraints given to member threads, timing constraints for the TMO.p itself are also given for non-overlapped computing. They are period, duration, and initial-offset. Timing constraints for all TMO.p's must be arranged so that there is no overlap of TMO.p's, by using a schedulability analyzer [Kim H. J. et al., 2008]. In this TMO.p scheme, real-time member threads of a TMO.p can be executed only when the TMO.p is in its duration. So, an SvM is suspended if the duration of a TMO.p is over, even though the message processing is not yet finished.

2.2. RT-eCos.p

RT-eCos.p [Lee et al., 2012; Kim J. G. et al., 2009] is a kernel that has been developed based on the open source eCos3.0, to support the partition computing of TMO.p. It provides a two-level real-time scheduler named Watch-dog Real-time Scheduler (WRS). The high-level scheduler is the partition scheduler responsible for on-time activation and deactivation of TMO.p partitions. The bottom-level task scheduler schedules the member threads (SpMs and SvMs) of a TMO.p in a deadline-driven manner, only when the TMO.p is active. The task-level scheduler supports task-based Earliest Deadline First (EDF) and Least Laxity First (LLF) policies. The functions of RT-eCos.p can be summarized as follows:

- On-time activation and deactivation of a TMO.p
- On-time invocation of an SpM and activation of an SvM by an event message
- Inter-partition and inter-task communication
- Deadline driven scheduling policies (EDF, LLF) for SpMs and SvMs

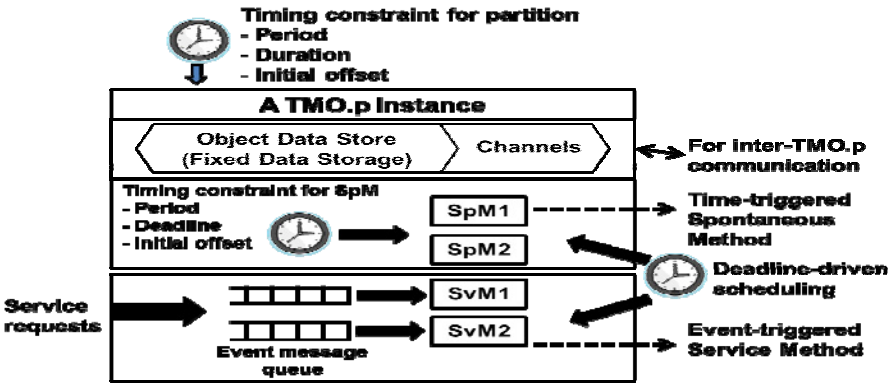

Fig. 1. The structure of a TMO.p instance

3. TMO.p-based S/W Fault-tolerant OFP System

An unmanned helicopter receives commands on flight mode from the GCS, and then the OFP calculates the values of control signals to be sent to control points, with the current sensor values from the GPS/INS, Altitude Heading Reference System (AHRS) and Helicopter Servo Actuator Switching Module (SWM). The four control points are cyclic-lateral, cyclic-longitudinal, collective, and tail rotor. Five basic auto-flight modes are take-off, hovering, auto-landing, point-navigation and multi-point-navigation.

In the TMO.p-based fault-tolerant OFP system, the OFP is dualized into one primary partition and one backup partition, as shown in Fig. 2. It takes over the following five kinds of faults:

- Excess of limitation in incremental changes of control-point values (Fail-notice fault)
- Deadline violations of time-triggered (SpMs) and message-triggered tasks (SvMs)
- Crash of the primary OFP (Fail-silent fault)
- Bad aircraft status in altitude, pitch, roll or yaw, due to wrong operations of the primary OFP (Byzantine faults)

The primary OFP partition uses a more progressive flight-control algorithm under refinement, and the backup OFP uses a safety-proven algorithm. The primary OFP aims at fast maneuvering and ascending/descending of an unmanned helicopter by enlarging the upper-limits and lower-limits of changes in control values.

Each of the primary and backup OFP partition consists of an Object Data Store (ODS), a Guide-and-Control SpM (GnC-SpM), a Health-Monitor-SvM and a GCS-reader task. The ODS stores the current values of AHRS, GPS, control points, aircraft capability parameters, and flight mode. Each GnC-SpM of the primary and backup OFP periodically calculates the next values of control points, after reading of the sensor values. The primary GnC-SpM wakes up the primary Health-Monitor-SvM, to do acceptance test, and system check at the end of its execution. According to the results of the acceptance test and system check, the primary Health-Monitor-SvM sends a success/fail message to the backup Health-Monitor-SvM, as a heart-beat. The backup GnC-SpM wakes up the backup Health-Monitor-SvM, to decide whether taking-over is necessary.

There are some differences in task structuring schemes of two OFP partitions as follows:

- The GCS-reader task of the primary is organized into a message-triggered thread (SvM), so that it can be activated as soon as it receives a command from the GCS. This is better for faster responses to the GCS, but it increases uncertainty of scheduling, because the activated GCS-reader SvM may disturb the execution of the GnC-SpM. However, the GCS-reader task of the backup is organized into a time-triggered task (SpM) which uses a non-blocked polling I/O. This scheme has a disadvantage in providing faster responses, but it has a benefit of increasing the predictability of scheduling.
- The scheduling policies of all member threads of the primary TMO.p are set to Earliest Deadline First (EDF) policy. This is also for fast responses. But the EDF policy may increase uncertainty of scheduling ,due to task preemptions. However, the scheduling policies of all member threads of the backup are set to First Triggered First Scheduled (FTFS) policy without preemption. This is to support timeliness guaranteed computing by increasing predictability. To use the FTFS policy, all threads (SpMs) are serialized so as to avoid overlaps of executions, by using the task serializer of [Kim H. J., 2008].

The followings are the major action sequences of the primary and backup Health-Monitor-SvMs:

- Primary Health-Monitor-SvM

1. Wait for receiving the calculation results from the primary GnC-SpM.
2. Perform acceptance-test with the received results and aircraft status.
3. Check the deadline violations of the system.
4. At success, send a success message to the backup Health-Monitor, and do external output to the aircraft and the GCS.

5. At fail, send a fail message to the Health-Monitor, and deactivate the primary OFP partition.

• Backup Health-Monitor SvM

1. Wait for receiving the calculation results from the backup GnC-SpM.

2. If (primary-active) then wait, with 1 milli-sec timeout, for a success/fail message from the primary Health-Monitor-SvM.

1) At timeout or fail (fail-silent fault, fail-notice fault), reset the primary-active flag and do external output to the aircraft and to the GCS.

2) At success, check the status of the aircraft to cover the Byzantine faults. If (fail) then reset the primary-active flag, disable the primary OFP partition, and do external output.

3. Else do external output, because the primary OFP partition has already been disabled.

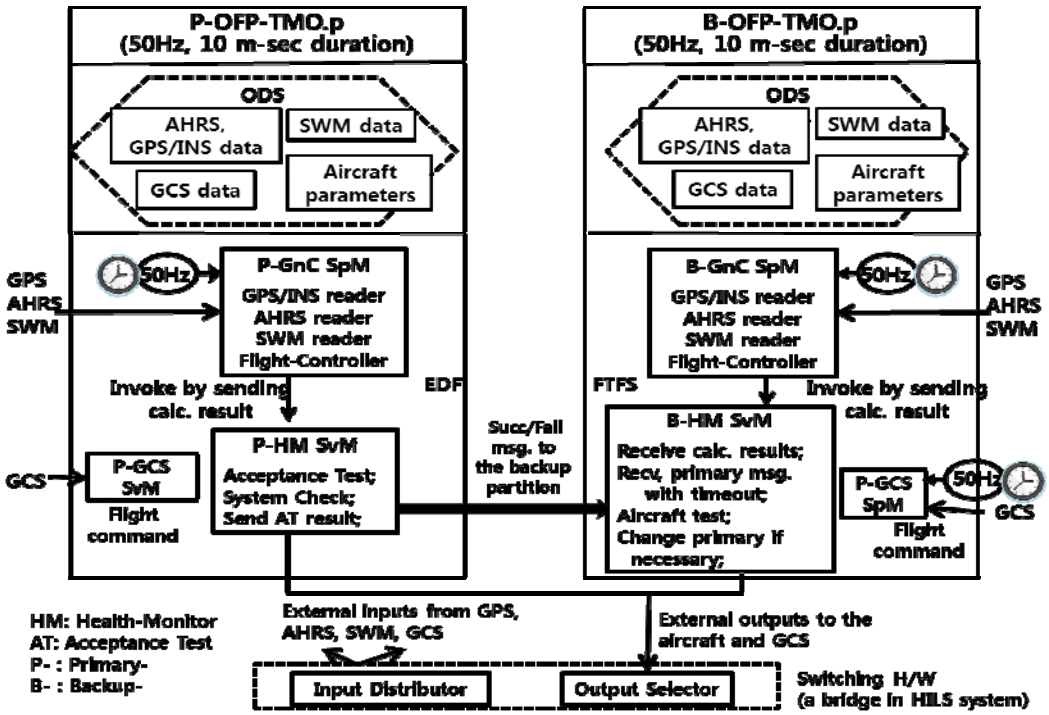

Fig. 2. The structure of the partition-based S/W fault-tolerant OFP

The timing constraints of the two partitions are given, so as to use non-overlapping CPU slots (Table 1). The period of each GnC-SpM in primary and backup is given, so that each SpM runs once in the duration of each partition. The deadlines of the SpMs and SvMs, shown in Table 2, are set by considering their Worst Case Execution Times (WCET).

Table 1. Timing constraints of the primary and backup OFP partitions

Timing constraints	Primary OFP partition	Backup OFP partition
Period	20 milli-sec (50Hz)	20 milli-sec (50Hz)
Duration	WCET of the primary OFP	WCET of the backup OFP
Initial offset	0	Duration of the primary OFP

Table 2. Timing constraints of the member tasks of each partition

Timing constraints	P-GnC-SpM/ B-Gnc_SpM	P-HM-SvM/ B-HM-SvM	P-GCS-SvM/ B-GCS-SpM
Period	20 milli-sec (50Hz)	none	None/ 50Hz
Deadline	8 milli-sec	1 milli-sec	1 milli-sec
Scheduling	EDF/FTFS	EDF/FTFS	EDF/FTFS

4. Experiments

To verify the robustness and timeliness of the S/W fault-tolerant OFP in TMO.p scheme, a HILS system has been used with the FlightGear-v0.9.10 simulator (Fig. 3). In the HILS environment, the primary and backup OFP receive information on GPS/INS, AHRS and SWM, and send the control signals from/to the FlightGear simulator. A bridge that connects the FCC with the FlightGear and the GCS has been developed as an input/output multiplexor. For the FCC, a board with an ST Thomson DX-66 STPC Client chip has been used. Fig. 4 shows the actual responses of the simulator and behaviors for fault task-over in point navigation mode. To show the fault task-over, an invented faulty scenario has been given to the HILS system. During 1 second, 20 consecutive faults with wrong control values have been generated. In the developed system, faults can be found earlier, but the acceptance test is inhibited, to make worse the attitude of the aircraft in this period. Before the last fault is injected, the acceptance test has been enabled to take over. As a result, the fault has been taken over by the backup OFP partition within about 11 milli-seconds, which is composed of the spare time to the WCET of the primary partition (about 1 milli-second), the worst-case scheduling latency of the RT-eCos.p (about 6 micro-seconds) and the WCET of the backup OFP partition (about 10 milli-sec) (Fig. 5).

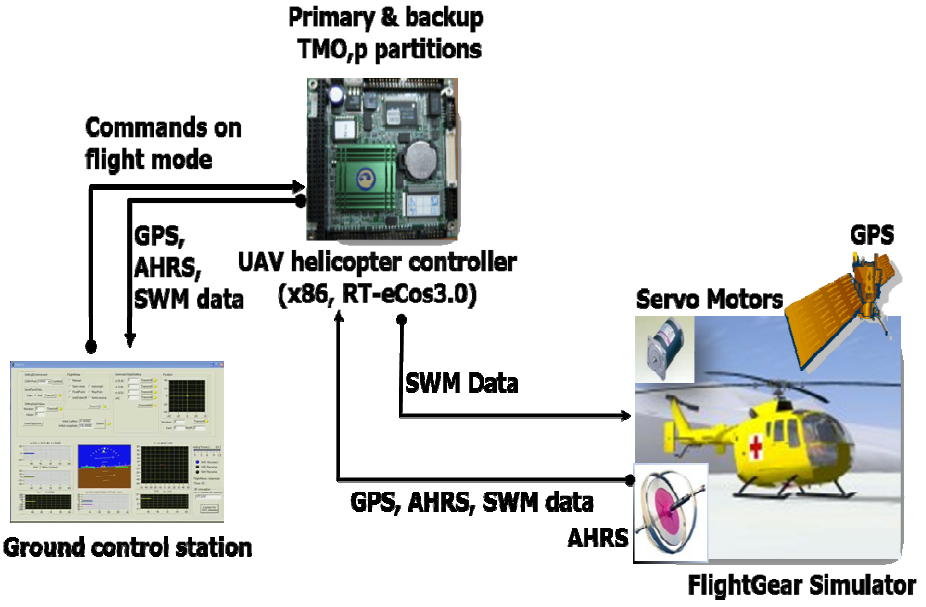

Fig. 3. The Hardware in-the-loop simulation system for an S/W fault-tolerant unmanned helicopter

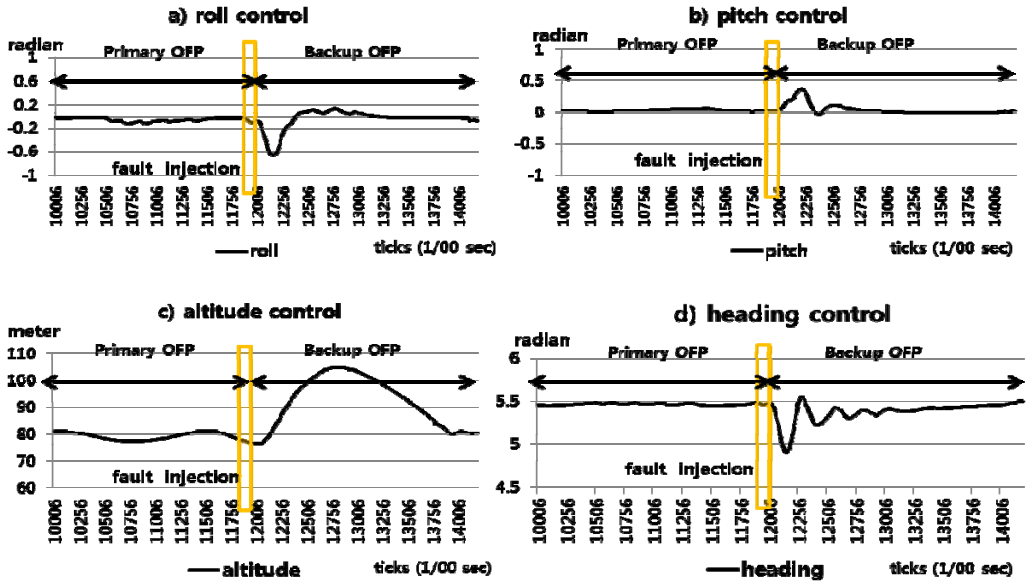

Fig. 4. (a) roll control, (b) pitch control, (c) altitude change, (d) heading control with fault-tolerance in point navigation mode

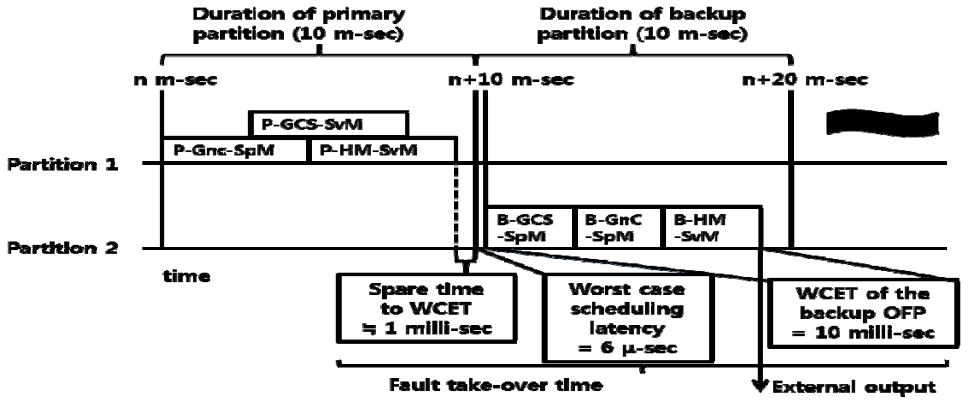

Fig. 5. Scheduling of the primary and backup partition and the fault take-over time

5. Conclusion

Fault-tolerant OFPs for UAVs have usually been implemented as primary-backup systems based on dual nodes. However, in the case of a small UAV, it is preferablle to implement a S/W fault-tolerant system that runs primary and recovery system together in a single FCC, to reduce the payload. For this case, a TMO.p-based S/W fault-tolerant system has been developed, to guarantee independent computing of the primary and backup OFP. The TMO.p-based approach is very much slimmer and faster, compared with the usual virtual

Machine approaches. Moreover, the TMO.p is a very suitable model for real-time computing, because it has time-triggered, and message-triggered, real-time threads as its members.

Acknowledgements

This work was supported by the Defense Acquisition Program Administration and Agency for Defense Development under contract.

References

[1] Kim, K. H., Kopetz, H. A real-time object model RTO.k and an experimental investigation of its potentials. 18th IEEE Computer Software & Applications Conference, pp. 392-402, IEEE Computer Society Press; 1994.
[2] Kim, S. G., Song, S. H., Chang, C. H., Kim, D. H., Heu, S., Kim, J. G. Design and implementation of an operational flight program for an unmanned helicopter FCC based on the TMO scheme. 7th IFIP WG 10.2 International Workshop, SEUS 2009, LNCS, vol. 5860, pp. 1-11, Springer, Newport Beach; 2009.
[3] Kim, D. H., Nodir, K., Chang, C. H., Kim. J. G. HELISCOPE project: research goal and survey on related technologies. 12th IEEE International Symposium on Object/component/service-oriented Real-time Distributed Computing, pp. 112-118, IEEE, Tokyo; 2009.
[4] Song, H. G., Kim, J. G., Heu, S. Design and implementation of a danger-aware operational flight program for an unmanned helicopter. 6th International Conference on Intelligent Unmanned Systems, pp. 9-15, ICIUS, Bali; 2010.
[5] Kim, J. G., Budiyono, A., Kim, D. M., Song, H. G., Kim, D. H. A TMO-based flight program of an unmanned helicopter. Aircraft Engineering and Aerospace Technology: An International Journal. vol. 83, pp. 353-362, Emerald; 2011 .
[6] Kim, K. H., Welch, H. O. Distributed execution of recovery blocks: an approach for uniform treatment of hardware and software faults in real-time applications. IEEE Transactions on Computers. vol. 38, pp. 626-636, IEEE; 1989.
[7] Kim, K.H., Bacellar, L., Subbaraman, C. Primary-shadow consistency issues in the DRB scheme and the recovery time bound. 7th International Symposium on Software Reliability Engineering, pp. 319 -329; 1996.
[8] Kim, J., Kim. D. H. Experimental analysis of primary-shadow replication scheme for fault-tolerant operational flight program of small scale UAV. 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, IEEE Computer Society Press, pp. 171-178, Newport Beach; 2011.
[9] Morgan, M.J. Integrated modular avionics for next-generation commercial airplanes. 4th Aerospace and Electronics (NAECON) IEEE National Conference, vol. 1, pp. 43-49; 1991.
[10] Airlines Electronic Engineering Committee: Avionics application software standard onterface part 1 - required/extended services. Aeronautical Radio Inc.; 2005.
[11] Han, S., Jin, H. W.: Full virtualization based ARINC 653 partitioning, 30th IEEE/AIAA Digital Avionics Systems Conference (DASC), pp.7E1-1-7E1-11, Sydney; 2011.
[12] VanderLeest, S. H. ARINC 653 hypervisor. 29th IEEE/AIAA Digital Avionics Systems Conference, pp.5.E.2-1-5.E.2-20, IEEE, Salt Lake City; 2010.
[13] Lee. J. T., Kim. J. G. TMO.p model and its scheduler for partition computing. J. Letters and Practices in Computing: A KISSE journal, vol. 18, pp. 733-741, KISSE; 2012.
[14] Kim, H. J., Kim, J. G.: An efficient task serializer for hard real-time TMO systems, 11th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, IEEE Computer Society Press, pp. 405-413, Orlando; 2008.
[15] Kim, J. G., Kim H. J., Park, J. H., Ju, H. T., Lee, B. E., Kim, S. G., Heu, S. TMO-eCos2.0 and its development environment for timeliness guaranteed computing. 1st Software Technologies for Dependable Distributed Systems, pp. 164-168, IEEE Computer Society Press, Tokyo; 2009.